Please note:

If you experience some difficulty in viewing some of the pages, use the magnifying tool to enlarge the specific section

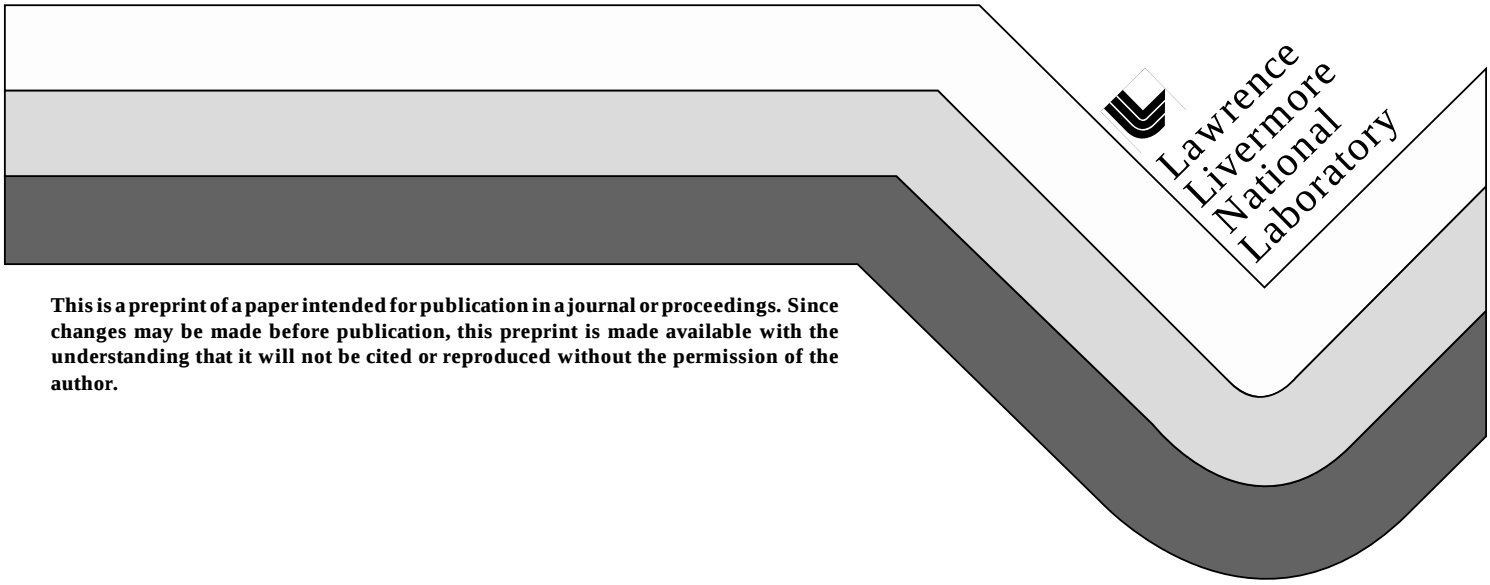
UCRL-JC-119206
PREPRINT

3D Mesh Optimization Methods for Unstructured Polyhedra: A Progress Report

Douglas S. Miller
Donald E. Burton

This paper was prepared for submittal to the
Eighth Nuclear Explosives Code Developers' Conference (NECDC)
Las Vegas, NV
October 25-28, 1994

November 22, 1994



3D Mesh Optimization Methods for Unstructured Polyhedra: A Progress Report

Douglas S. Miller, Donald E. Burton
Lawrence Livermore National Laboratory

Abstract

A mesh optimization scheme allows a Lagrangian code to run problems with extreme mesh distortion by reconfiguring node and zone connectivity as the problem evolves. We have developed some 3D mesh optimization operations and criteria for applying them. These are demonstrated in a 3D Free Lagrange code being developed at LLNL. In the simplest case of a mesh or mesh subregion composed purely of tetrahedra we can maintain a Delaunay tetrahedralization. For more interesting meshes, made up of general polyhedra, a suite of optimization operations and their respective application criteria have been developed.

Introduction

We are designing mesh optimization methods for a three-dimensional free-Lagrange hydrodynamics code. The advantage to the free-Lagrange method is that the mesh can be re-defined at will, which allows free-Lagrange codes to run problems that cannot be handled by a purely Lagrangian approach. We are developing a mesh optimization capability that will allow us to maintain a "good" mesh even in the case of extremely distorted flows. The work described here is being done in a code currently under development at Livermore. We are building a three dimensional, free-Lagrange hydrodynamics code that uses arbitrary polyhedra for zones. It uses a finite volume approach to compute solutions to the hydrodynamics equations [1]. This project is nowhere near completion, but we do have some early results to report.

Mesh optimization has three goals. First, tangled meshes must be avoided or repaired. For example, hourglassing can drive the timestep to zero or even turn a zone inside out. An optimization package should recognize that the zone is developing a problem and do something about it. Second, resolution can be increased where it is needed. More points can be added around areas of interest, e.g., a shock wave, a material interface, or a small feature in a large problem (a bolt in an I-beam, for example). The third goal is to reduce unnecessary computational effort lowering resolution in regions that do not need it.

The problem of defining a "good" mesh for a given problem has no unique solution. Several algorithms exist [6, 7] for defining a mesh but computing a new mesh for an entire problem by these methods is too computationally expensive to do frequently. Instead we have chosen to allow the mesh to flow with the fluid in a purely Lagrangian way until one or more zones becomes "bad" (we will define "bad" later). We then focus our attention on repairing the bad zones and the surrounding region.

Short Term Optimization Plan

We have two immediate mesh optimization goals; to change the connectivity of nodes of that make up bad zones so that we end up with a better mesh and to refine regions that need improved resolution. Later we will add the counterpart of refinement, coarsening the mesh.

When zones become bad, our current strategy is to decompose them into two or more tetrahedra. A tetrahedron, or "tet", is the simplest possible 3D zone. Dealing with bad tets is much easier than trying to handle the general case of a bad arbitrary polyhedron. The tetrahedra in a mesh are then reconnected such that all the tetrahedra are locally Delaunay[3] (which we will define below).

To improve resolution we have three mesh operations; add a point to a zone, add a point to a face, and add a point to an edge. The last two operations are currently defined for faces and edges that are common to tetrahedral zones only. It is possible to define these operations on arbitrary polyhedra but it is unclear how to do this in a way that gives predictable and useful results. Currently we use maximum volume and area criteria for our zone and face refinement operations. Edge refinement is controlled by both maximum length and aspect ratio criteria.

Operations

In this section we will discuss the details of our various mesh operations.

Tetrahedral Decomposition

Tetrahedral decomposition reduces a polyhedron to a collection of tets. This would be trivial if we added a point to the center of the zone to be decomposed, but that would add resolution and needlessly decrease the timestep. Instead, we want an algorithm that uses only the points that already make up the zone to be decomposed.

One might at first imagine that it would suffice to design a method that decomposes a small set of polyhedra (for example, hexahedra, prisms, and pyramids) but this is not so. Consider two adjacent hexahedra. Once a hexahedron, or "hex", has been decomposed, its faces are split into triangles and its neighbor now has seven faces, not six—the neighbor is no longer a hex at all but a seven sided polyhedron. A more general method is required.

Tetrahedral decomposition is accomplished with the following algorithm.

```
    Make_Bad_Zone_List
(1) Triangulate_Faces_of_Bad_Zones
    for each (Zone in Bad_Zone_List) do
       while (number_faces_in_zone > 4) do
(2)       Make_Point_to_Point_Chart
(3)       Try_to_Find_a_Tet
          if (tet_is_found) then
             Split_Tet_from_Zone
          else
(4)          Flip_a_Face_Triangle_Pair
          endif
       endwhile
    endfor
```

The steps that need some explanation are numbered.

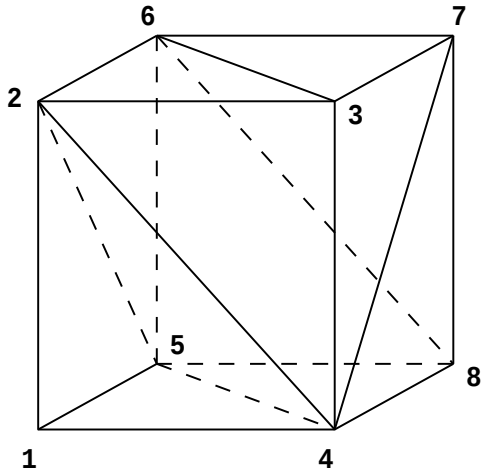


Figure 1: The faces of this zone have been split into triangles, in preparation for decomposing it into tetrahedra. The chart that represents this zone is shown below.

`Split_Faces_of_Bad_Zones_into_Triangles` examines each face of a zone and splits any face consisting of more than three points into triangles. In theory it is usually possible to split the faces such that tetrahedralization of the zone is trivial afterwards, but we have not found an inexpensive way to calculate the correct splitting in advance. It is simpler to just split the faces arbitrarily and adjust afterwards.

Once the faces are split into triangles, we can begin to look for tets. We make a chart of the connections between points. See Figure 1. This hex has been triangulated and the resulting chart is shown below. A one means "is connected", a zero means "is not connected".

```
      1 2 3 4 5 6 7 8

1     0 1 0 1 1 0 0 0
2     1 0 1 1 1 1 0 0
3     0 1 0 1 0 1 1 0
4     1 1 1 0 1 0 1 1
5     1 1 0 1 0 1 0 1
6     0 1 1 0 1 0 1 1
7     0 0 1 1 0 1 0 1
8     0 0 0 1 1 1 1 0
```

Step (3), `Try_to_Find_a_Tet`, is accomplished by examining this chart. For each row, we take the set of connected points and look for "rings". We define a ring to be three points that are all mutually connected. For example, in row 1, we have the set $\{2, 4, 5\}$. Let the symbol $\leftrightarrow$ mean "is connected to".

From the chart, $2 \leftrightarrow 4$, $4 \leftrightarrow 5$, and $5 \leftrightarrow 2$, hence these points make a ring. Point 1 is connected to every point in the ring ($1 \leftrightarrow \{2,4,5\}$), so point 1 and the set $\{2,4,5\}$ form a tetrahedron. Having found a tet, we pop the tet $\{1,2,4,5\}$ off the zone, create an updated chart of the zone and repeat this operation again.

It can happen that a point is not connected to any rings. In the chart above, point 3 is connected to the set $\{2,4,6,7\}$. None of the triplets in this set ($\{2,4,6\}$, $\{2,4,7\}$, $\{2,6,7\}$, $\{4,6,7\}$) are rings. If *none* of the points in the chart are connected to a ring, then we look for the next best thing; a "near-ring", in which one connection is missing. In our example hex, $\{4,2,6\}$ is a near-ring; $4 \leftrightarrow 2$, $2 \leftrightarrow 6$, but $6 \not\leftrightarrow 4$. This would show up when we examine point 5 because $5 \leftrightarrow \{4,2,6\}$. Assuming we had already searched the table for a ring without success, then having found the near-ring, we would go ahead and force the connection $6 \leftrightarrow 4$. Then we pop the new tet, $\{5,4,2,6\}$, off the zone. In practice we limit the number of new connections like this to one per zone, although for zones made up of many faces (greater than fifteen or twenty triangular faces, say) more than one new connection might be helpful. Limiting ourselves to just one new connection per zone eliminates the need to ensure that newly connected tets do not intersect each other. If we ever do calculations that develop such many-sided zones, it will be straightforward to add a tet intersection check and make more than one new connection during a decomposition.

As a practical measure, some tets generated by our decomposition algorithm have to be tested for small or zero volumes lest the new connection algorithm in step `(3)` create logical tetrahedra that are degenerate (all the points lie in a plane, *e.g.* $\{1,2,3,4\}$) or "wafer thin". We test that the *sine* of the angle between two zone faces that will be part of the new tet created by a new connection is at least 0.2. This prevents tets that have extremely small, zero, or (even worse) negative volume from forming.

Even having made a new connection through the interior of a zone, it can happen that there are no tets to be found. This situation can arise immediately after the faces have been split into triangles or after several tetrahedra have been removed from the original zone. A common instance is the "evil prism", shown in Figure 2a. The only way out of this problem is to flip one or more of the triangular face pairs. Once such a flip is made, the algorithm can continue. Here we have flipped $5 \leftrightarrow 2$ to $1 \leftrightarrow 6$, and the prism immediately falls apart into the tets $\{1,4,6,5\}$, $\{1,6,3,2\}$, $\{1,6,4,3\}$.

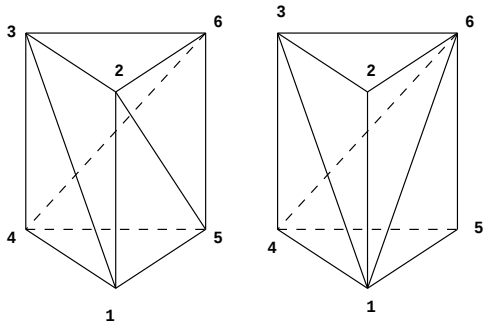


Figure 2: (a) an un-tetrahedralizable prism, (b) after a face pair flip this prism decomposes immediately into three tets

Delaunay Tetrahedralization

Once a region of the mesh has been broken into tets, we reconnect the tets such that all "interior tet faces" are locally Delaunay. An "interior tet face" is a face that is common to two adjacent tets. The criterion for being locally Delaunay is that the circumsphere of one tet does not enclose the other. The process of how to make any tetrahedral mesh Delaunay is described in detail by others[5, 4]. In brief, three tetrahedral flips are necessary.

Given two adjacent, non-Delaunay tets $\{a,b,c,d\}$ and $\{a,b,c,e\}$ that form a convex hull, the two tets are reconnected into three; $\{a,c,d,e\}$, $\{a,b,d,e\}$, and $\{b,c,d,e\}$. This is called the "two to three flip".

For the case that $\{a,b,c,d\}$ and $\{a,b,c,e\}$ do not form a convex hull, one must check to see if a third tet exists that combined with the first two completes a convex hull. Assume (given the two tets $\{a,b,c,d\}$ and $\{a,b,c,e\}$) that the $\{a,b,d\}$ and $\{a,b,e\}$ faces form a concave surface and the tet $\{a,b,d,e\}$ exists. Then the three tets $\{a,b,c,d\}$, $\{a,b,c,e\}$ and $\{a,b,d,e\}$ are transformed to the two tetrahedra $\{a,c,d,e\}$ and $\{b,c,d,e\}$. This is the "three to two flip".

One more flip is necessary to handle degenerate situations. Consider four tets $\{a,c,d,f\}$, $\{b,c,d,f\}$, $\{a,c,f,e\}$, and $\{b,c,e,f\}$. The four to four flip is used to fix the case of $\{a,b,c,f\}$ being co-planar points. Essentially, the $\{c,f\}$ connection is replaced by $\{a,b\}$, to give four new tets $\{a,b,c,d\}$, $\{a,b,d,f\}$, $\{a,b,c,e\}$, and $\{a,b,e,f\}$. A "two to

two" version of the four to four flip is necessary to maintain good surface triangulation; it is identical to the four to four flip but only the tets $\{a, c, d, f\}$ and $\{b, c, d, f\}$ are involved and the faces $\{a, c, f\}$ and $\{b, c, f\}$ must lie on a surface boundary of the mesh region.

Provided that the Delaunay algorithm operates by means of local transformations (flips), it is simple to modify a Delaunay tetrahedralization algorithm to work in a heterogeneous mesh (a mesh composed of more zone types than just tetrahedra). Instead of examining all the faces in the mesh for the Delaunay criterion, only the faces that are common to two tetrahedra are considered. The result is a method which allows a mesh of arbitrary types of polyhedra but keeps the tetrahedral regions of the mesh Delaunay.

Refinement

After eliminating tangled zones, the next most important feature of mesh optimization is the ability to refine the mesh in selected regions. We can increase resolution by adding more zones, edges or points, depending on the kind of resolution we want to improve (this is a one way street; zones can be added without adding new edges or points but to add a new point one must necessarily add new edges and faces). Although this fine level of control is interesting to consider, in practice it is not clear that significantly better accuracy can be obtained by splitting zones without adding points, since in a staggered grid hydrodynamics scheme (which is our current method) the momentum control volume is centered on the points, not the zone center. The momentum control volume might change shape as zones are fragmented, but it will not get significantly smaller unless additional points are added. As a result, we have only implemented refinement operations that add points to the mesh.

We have implemented three zone splitting refinement options; edge-centered, face-centered, and zone-centered. Edge and face-centered zone splits are defined only for tetrahedra, zone-centered splitting is defined for all convex zones.

In a zone-centered zone split operation, a new point is added to the center of the zone to be split. A pyramid is constructed from each face and the zone center point (*e.g.*, for a cube one would obtain six square pyramids, whereas a triangular prism would yield three square pyramids and two tetrahedra).

Face-centered splits are done only on tetrahedra because higher order polyhedra would become concave under this operation. Consider a tet, $\{a, b, c, d\}$. A point e is added to the center of the face $\{a, b, c\}$. The face is split by the new edges $\{a, e\}$, $\{b, e\}$, and $\{c, d\}$, and the connection $\{d, e\}$ is made so that in the end we have three new tets, $\{a, b, e, d\}$, $\{b, e, c, d\}$, and $\{c, e, a, d\}$.

Edge-centered zone splits are also defined only on tetrahedra. Consider once again a tet, $\{a, b, c, d\}$. A point e is added to the center of the edge $\{a, b\}$. The original tet is split into two new ones, $\{a, e, c, d\}$ and $\{b, c, e, d\}$. The edge split is especially useful for maintaining good tet aspect ratios.

Criteria

Teaching a computer when and how to modify a mesh during a time dependent problem is still best described as a "dark art". Our current approach is to try to maintain aspect ratios as close to unity as we can, and to keep a set level of resolution by keeping zone volumes, face areas, and edge lengths all within user prescribed limits.

For many problems the best mesh for finite-difference and finite volume techniques is orthogonal and evenly spaced, hence one measure of zone quality is how far from being orthogonal the faces of the zone are. We measure this by finding the angle between a vector normal to a face and a vector defined by the zone center and the face center. When this angle is larger than a user defined maximum, we perform a tetrahedral decomposition on the zone.

The Delaunay criterion for the tetrahedral regions of the mesh yields a good tet mesh for the most part but it also allows some "wafer thin" tets at the boundaries of the tet regions. A criterion to detect these and an operation to eliminate them must be added to our package.

Examples

Figure 3 shows our current optimization package operating on a 5x5x5 hexahedral mesh to which has been applied a Coggeshall velocity field. The velocity $\mathbf{u}$ of a point is given (in spherical coordinates r, θ, ϕ) by [2]

$$u_r = \frac{r}{2(t-1)} \quad (1)$$

$$u_\theta = 0 \quad (2)$$

$$u_\phi = \frac{r}{(t-1)}\sqrt{4\left(\frac{\gamma-1}{\gamma}\right)\sin^3\theta + \frac{1}{12}} \quad (3)$$

where t is the time and $\gamma = 5/3$. Note that as the hex zones become distorted they are broken into

tetrahedra. Once created, the tetrahedra maintain the Delaunay relation. Refinement was turned off for this test, so no new points are added even when the zones become large and extremely distorted.

Our second example is an analytic velocity field that emulates the fluid motion of an unstable boundary layer applied to the same 5x5x5 hexahedral grid as above. In Figure 4 face refinement only takes place within the mesh. It is easy to see the need for edge refinement as the zone aspect ratios become large.

Future Work

Our mesh optimization package is far from finished. Much research remains to be done to correctly match control criteria with mesh operations. In the future we will examine the effects of adding information about the current solution to the mesh optimization routines. Additional mesh operations will be added. A zone merge operation to combine and eliminate "wafer thin" tetrahedra is necessary. More refinement and reconnection operations on non-tetrahedral zones are needed if we are to move beyond our current strategy of tet decomposition when zones develop undesirable shapes.

Acknowledgments

We would like to gratefully acknowledge the advice and support of Dave Lappa and Todd Palmer, both of LLNL, in this effort. This work was performed under the auspices of the U.S. Department of Energy, by Lawrence Livermore National Laboratory under Contract #W-7405-Eng-48.

References

[1] D.E. Burton. Multidimensional discretization of conservation laws for unstructured polyhedral grids. Technical Report UCRL-JC-118036, Lawrence Livermore National Laboratory, August 1994.

[2] S.V. Coggeshall and J. Meyertervehn. Group-invariant solutions and optimal systems for multidimensional hydrodynamics. *Journal of Mathematical Physics*, 33(10):3585–3601, Oct 1992.

[3] B. Delaunay. *Izv. Akad. Nauk. SSSR Math and Nat. Sci. Div.*, 6(793), 1934.

[4] Dan Fraser. Tetrahedral meshing considerations for a three-dimensional free-lagrangian code. Technical Report LA-UR-88-3707, Los Alamos National Laboratory, 1988.

[5] Barry Joe. Three-dimensional triangulations from local transformations. *SIAM J. Sci. Stat. Comput.*, 10(4):718–741, July 1989.

[6] Joe F. Thompson, Z.U.A. Warsi, and Wayne C. Martin. *Numerical Grid Generation*. North Holland, New York, 1985.

[7] N.P. Weatherill and G. Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37:2005–2039, 1994.

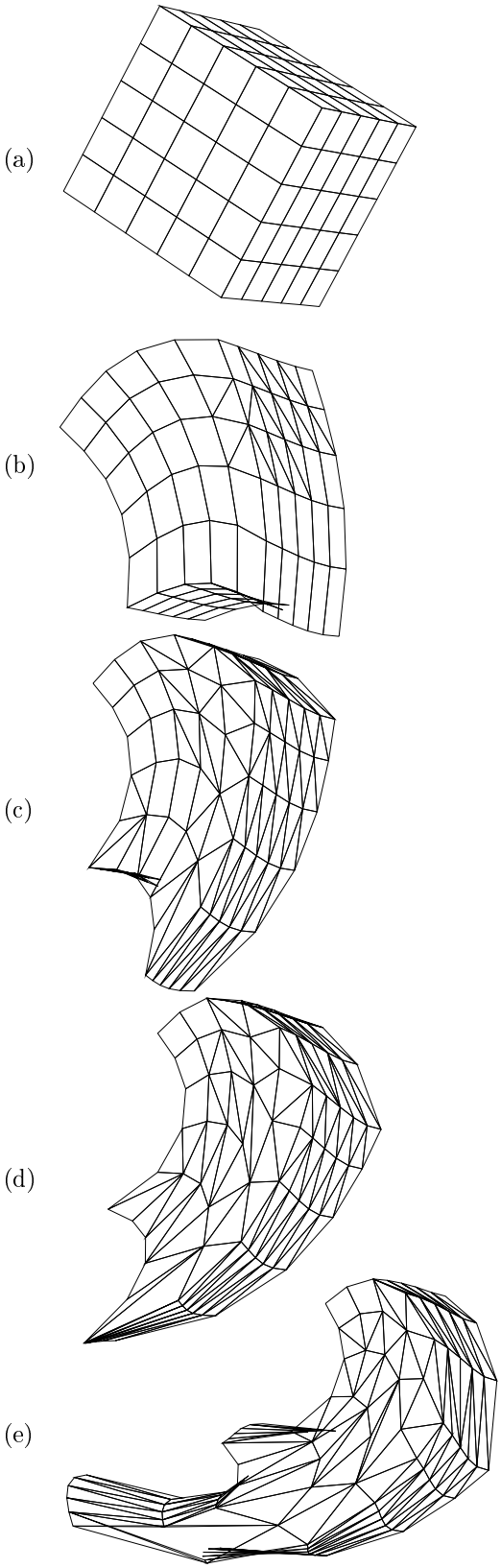


Figure 3: Coggeshall velocity field applied to a 5x5x5 hex mesh at times (a) 0.0, (b) 0.5, (c) 0.7, (d) 0.8, and (e) 0.9

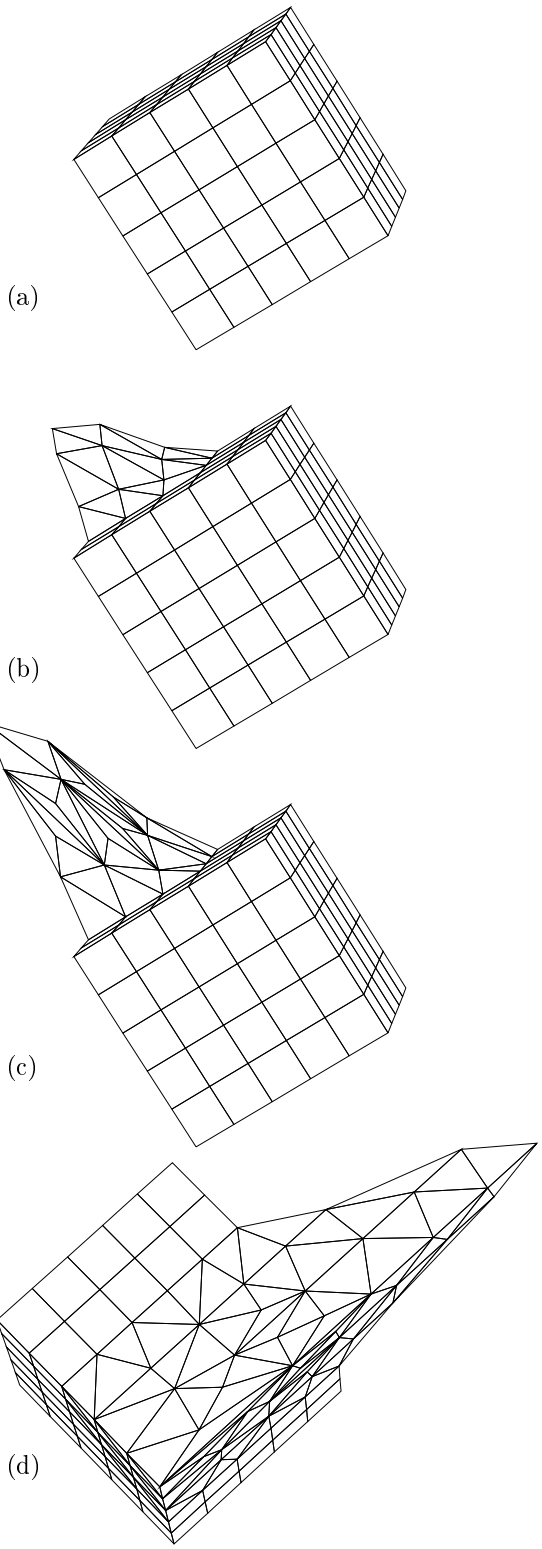


Figure 4: Instability velocity field applied to a 5x5x5 hex mesh at times (a) 0.0, (b) 0.6, (c) 1.2, (d) 1.2, side view